MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1 2018/2019

TSE3151/TSD2711 – SOFTWARE DESIGN

(All Sections/Groups)

13 October 2018
09:00 AM – 11:00 AM
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. The total number of pages for this examination paper is fifteen(15) pages including the cover page.

2. This examination paper contains a total of six(6) questions divided into two(2) different sections, Section A and Section B, respectively. Each section contains three(3) questions.

3. You are required to answer a total of four(4) questions. You must answer any two(2) questions from Section A, and any two(2) questions from Section B.

4. Each question carries an equal score of 20 points. The total score for this examination paper is 80 points.

SECTION A

(Answer any two(2) questions in this section)

QUESTION A1

A1(1) Design begins from requirements. Discuss the consequences of the following issues on software design:

(a) Incorrect requirements.
(b) Inconsistent requirements.
(c) Incomplete requirements.

[3 x 2 points = 6 points]

A1(2) Describe one scenario for each of the following design patterns. You may provide examples in your answers.

(a) singleton design pattern.
(b) composite design pattern.
(c) readers-writers lock design pattern.

[3 x 2 points = 6 points]

A1(3) Draw an appropriate class diagram for the requirements written in italics below. Label the diagram(s) appropriately. Include foreign keys (FK) where appropriate.

An excerpt from the Satellite Vehicle Tracking (SVT) system requirements.

Reference item (6.2.2): The raw data record structure provided to the SVT database from the Satellite Black Box (SBB) transmission is as follows:

(a) SBB ID	*varchar(10);*
(b) Vehicle ID	*varchar(10);*
(c) Vehicle destination	*varchar(50);*
(d) Data date	*date;*
(e) Data time	*time with time zone;*
(f) Latitude	*double;*
(g) Longitude	*double;*
(h) SBB Message	*varchar(50);*

[Total = 8 points]

Continued .....

QUESTION A2

A2(1) Provide three(3) reasons why software requirements must be documented.

[3 x 2 points = 6 points]

A2(2) Answer the following questions. You may provide examples in your answers.

(a) Explain the meaning of design having a focus on concurrency.

[2 points]

(b) Describe the conceptual differences between data abstraction and control abstraction in software design.

[2 points]

(c) Discuss two(2) design issues regarding control and handling of events in system and software design.

[2 points]

A2(3) Draw an appropriate activity diagram as the model behavioural design element to fulfill the requirements written in italics below. Label the diagram(s) appropriately.

An excerpt from the Satellite Vehicle Tracking (SVT) system requirements. This question is related to the data structure in question A1(3) above. Reference item (6.2.2): Vehicle Location Data Validation

(6.2.3) The SVT system must validate and verify that each raw data record received comply with the data structure in reference item 6.2.2.

[2 points]

(6.2.4) In the event that any field data does not comply, the entire record shall be considered corrupt.

[2 points]

(6.2.5) A separate register of all corrupted records must be maintained in the system. This register will be used to track "faulty SBB" or "communications failure between the SBB and the SVT database".

[2 points]

(6.2.6) The target is to separate the good SVT data and the corrupted SVT data into two different data tables.

[2 points]

[Total = 8 points]

Continued ...

QUESTION A3

A3(1) Answer the following questions. You may provide examples in your answers.

(a) Explain the differences between explicit, implicit and emergent requirements in software design.

(b) What is the primary purpose of database normalization in data-centric architectural design?

(c) Describe the concept of Single-Responsibility Principle (SRP) in object-oriented design.

[3 x 2 points = 6 points]

A3(2) Answer the following questions. You may provide examples in your answers.

(a) Provide two(2) implementation issues that object-oriented design overcome the weaknesses in procedural or imperative software design.

(b) The Service-Oriented Architecture (SOA) design makes it easy for computers connected over a network to cooperate. Describe one(1) example of a use-case scenario implementation of the SOA design.

(c) What does design using standardized component based architectures mean? Provide two(2) examples of standardized software components.

[3 x 2 points = 6 points]

A3(3) The following diagram illustrates the facade design pattern.

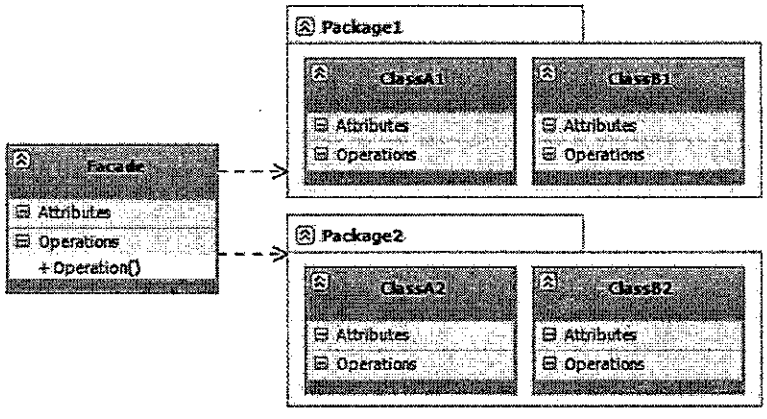

Fig A3.3 The Facade Design Pattern

Describe the *Concepts, Principles, Methods and Tools* in the implementation of the facade design pattern.

[4 x 2 points = 8 points]

Continued .....

SECTION B

(Answer any two(2) questions in this section)

QUESTION B1

B1(1) Answer the following questions. You may provide examples in your answers.

(a) "*A state diagram may have a transition event that returns to itself.*" What does this statement mean?

(b) "*Compiled applications are targeted towards a particular platform and so are platform independent.*" Discuss this statement.

(c) Discuss two(2) reasons why we must take advantage of different features provided in different programming languages in our software design.

[3 x 2 points = 6 points]

B1(2) Answer the following questions. You may provide examples in your answers.

(a) Why is it important to consider design quality for maintainability? Provide two(2) reasons.

(b) What is the Scheduled-Task design pattern? When is it appropriate to use it?

(c) Design for security is about preventing the outside world from harming the system while design for safety is about preventing the system from harming the outside world. Discuss two(2) examples on how we can implement safety in software design.

[3 x 2 points = 6 points]

B1(3) Answer the following questions based on the example software code provided on the next page

(a) Describe the concept of factory design pattern in general.
(b) Describe the principles involved in factory design pattern.
(c) From the code, explain the method used in factory design pattern.
(d) Are there any specific tools or features required to implement factory design pattern? Explain.

[4 x 2 points = 8 points]

Continued ....

Source code for Question B1(3)

```
// C++ PROGRAM TO DEMONSTRATE FACTORY DESIGN PATTERN

#include <iostream>
using namespace std;

enum VehicleType {
  VT_TwoWheeler,  VT_ThreeWheeler,  VT_FourWheeler
};

// LIBRARY CLASSES ==========
class Vehicle {
  public:
    virtual void printVehicle() = 0;
    static Vehicle* Create(VehicleType type);
  };
  class TwoWheeler : public Vehicle {
  public:
    void printVehicle() {
        cout << "I am two wheeler" << endl;
    }
  };
  class ThreeWheeler : public Vehicle {
  public:
    void printVehicle() {
        cout << "I am three wheeler" << endl;
    }
  };
  class FourWheeler : public Vehicle {
    public:
    void printVehicle() {
        cout << "I am four wheeler" << endl;
    }
  };

// FACTORY DESIGN PATTERN METHOD ==========
Vehicle* Vehicle::Create(VehicleType type) {

    if (type == VT_TwoWheeler)
        return new TwoWheeler();
    else if (type == VT_ThreeWheeler)
        return new ThreeWheeler();
    else if (type == VT_FourWheeler)
        return new FourWheeler();
    else return NULL;
  }
```

Continued ...

```
// CLIENT CLASS ==========
class Client {
        public:
          Client()
          {
                VehicleType type = VT_ThreeWheeler;
                pVehicle = Vehicle::Create(type);
          }
          ~Client() {
                if (pVehicle) {
                  delete[] pVehicle;
                  pVehicle = NULL;
                }
          }
          Vehicle* getVehicle()  {
                return pVehicle;
          }
        private:
          Vehicle *pVehicle;
        };

// DRIVER PROGRAM ==========
int main() {
        Client *pClient = new Client();
        Vehicle * pVehicle = pClient->getVehicle();
        pVehicle->printVehicle();
        return 0;
}
// ==========
```

Fig: Source code for Question B1(3)

Continued....

QUESTION B2

B2(1) Answer the following questions. You may provide examples in your answers.

(a) Describe two(2) differences between sequential and parallel designs of software applications.

(b) Provide two(2) design considerations between software that runs on stack memory and software that runs on heap memory.

(c) Discuss time-sharing and multi-tasking concepts in software design.

[3 x 2 points = 6 points]

B2(2) Answer the following questions. You may provide examples in your answers.

(a) How is the Message Passing Interface (MPI) library used in parallel and distributed processing? Explain how it can be used for local and distributed executions.

(b) "*The Null Object design pattern is a useful construct in software application design.*" Describe one(1) example where we implement this design pattern.

(c) "*In object-oriented programming, object serialization and de-serialization are about abstract objects data transfer.*" Clarify what this statement means.

[3 x 2 points = 6 points]

B2(3) Answer the following questions based on the example software code provided on the next page.

(a) Describe the concept of thread-pool design pattern in general.
(b) Describe the principles involved in thread-pool design pattern.
(c) From the code explain the method used in thread-pool design pattern.
(d) Are there any specific tools or features required to implement thread-pool design pattern? Explain.

[4 x 2 points = 8 points]

Continued....

Source code for Question B2(3)

```
// C++ PROGRAM TO DEMONSTRATE THREAD-POOL DESIGN PATTERN

#include <thread>
#include <mutex>
#include <condition_variable>

class ThreadPool;

// WORKER THREAD CLASS
// ==========
class Worker {
public:
        Worker(ThreadPool &s) : pool(s) { }
        void operator()();
private:
        ThreadPool &pool;

void Worker::operator()() {
        std::function<void()> task;

        while(true) {
                {
                // ACQUIRE LOCK
                std::unique_lock<std::mutex> lock(pool.queue_mutex);

                // LOOK FOR A WORK ITEM
                while(!pool.stop && pool.tasks.empty()) {

                        // IF NO WORK ITEM WAIT FOR NOTIFICATION
                        pool.condition.wait(lock);
                }
                // EXIT IF POOL IS STOPPED
                if(pool.stop)  return;

                // GET THE TASK FROM THE QUEUE
                task = pool.tasks.front();
                pool.tasks.pop_front();

                }  //RELEASE LOCK

                // EXECUTE THE TASK
                task();

        } // END WHILE... TRUE

} // END void Worker::operator()()

}; // END class...Worker
```

Continued ...

```
// THE THREAD-POOL CLASS
// ==========================================
class ThreadPool {
public:
        ThreadPool(size_t); // CONSTRUCTOR
        template<class F>
        void enqueue(F f);
        ~ThreadPool();      // DESTRUCTOR
private:
        friend class Worker;

        // TO KEEP TRACK OF THREADS
        std::vector< std::thread > workers;

        // THE TASK QUEUE
        std::deque< std::function<void()> > tasks;

        // THE SYNCHRONIZATION
        std::mutex queue_mutex;
        std::condition_variable condition;
        bool stop;

// THE THREAD POOL CONSTRUCTOR
ThreadPool::ThreadPool(size_t threads)
  : stop(false)
{
        for(size_t i = 0;i<threads;++i)
                workers.push_back(std::thread(Worker(*this)));
}

// THE THREAD POOL DESTRUCTOR
ThreadPool::~ThreadPool()
{
        // STOP ALL THREADS
        stop = true;
        condition.notify_all();

        // JOIN ALL THREADS
        for(size_t i = 0;i<workers.size();++i)
                workers[i].join();
}

// ADD NEW WORK ITEM TO THE POOL
template<class F>
void ThreadPool::enqueue(F f)
{
  {
    // ACQUIRE LOCK
    std::unique_lock<std::mutex> lock(queue_mutex);
```

Continued ...

```
        // ADD THE TASK
        tasks.push_back(std::function<void()>(f));

    } // RELEASE LOCK

    // WAKE UP ONE THREAD
    condition.notify_one();
}

}; // END class ... ThreadPool
```

Fig: Source code for Question B2(3)

Continued ...

QUESTION B3

B3(1) Answer the following questions. You may provide examples in your answers.

(a) What is polymorphism principle in the context of software design? Describe by giving an example how polymorphism is implemented in object-oriented design.

(b) "*Object-oriented design implements nonlinear logic and not serial logic.*" Discuss this statement.

(c) Provide two(2) different scenarios, one each for circumstances where we use the ***while(expression) .. loop*** and the ***do .. until(expression) loop.***

[3 x 2 points = 6 points]

B3(2) Answer the following questions. You may provide examples in your answers.

(a) "*The Domestic Washing Machine is an appropriate platform to implement the Event-Driven design architecture.*" Provide two(2) reasons for it being appropriate.

(b) "*The Apache Cassandra database is said to be data fault-tolerant.*" Explain what this statement means.

(c) The Gnuplot software component is a software component in the true sense. Discuss how the command design pattern is used when interfacing the Gnuplot component with python programs.

[3 x 2 points = 6 points]

B3(3) Answer the following questions based on the example software code provided on the next page.

(a) Describe the concept of memento design pattern in general.
(b) Describe the principles involved in memento design pattern.
(c) From the code explain the method used in memento design pattern.
(d) Are there any specific tools or features required to implement memento design pattern? Explain.

[4 x 2 points = 8 points]

Continued....

Source code for Question B3(3)

```
// C++ PROGRAM TO DEMONSTRATE MEMENTO DESIGN PATTERN

#include <iostream.h>
class Number;

// THE MEMENTO CLASS
class Memento
{
public:
        Memento(int val) {
        _state = val;
        }

private:
        friend class Number;
        int _state;
}; // END class ... Memento

// THE NUMBER CLASS
class Number
{
public:
        Number(int value) { _value = value; }
        void dubble() { _value = 2 * _value; }
        void half() { _value = _value / 2; }
        int getValue() { return _value; }

        Memento *createMemento() {
                return new Memento(_value);
        }

        void reinstateMemento(Memento *mem) {
                _value = mem->_state;
        }

private:
        int _value;

}; // END class ... Number
```

Continued ...

```
// THE COMMAND CLASS
class Command
{
public:
        typedef void(Number:: *Action)();

        Command(Number *receiver, Action action) {
                _receiver = receiver;
                _action = action;
        }

        virtual void execute() {
                _mementoList[_numCommands] = _receiver->createMemento();
                _commandList[_numCommands] = this;
                if (_numCommands > _highWater) {
                        _highWater = _numCommands;
                }
                _numCommands++;
                (_receiver-> *_action)();
        }

        static void undo() {
                if (_numCommands == 0) {
                        cout << "*** Attempt to run off the end!! ***" << endl;
                        return ;
                }
                _commandList[_numCommands - 1]->_receiver->reinstateMemento
                        (_mementoList[_numCommands - 1]);
                _numCommands = _numCommands -1;
        }

        void static redo() {
                if (_numCommands > _highWater) {
                        cout << "*** Attempt to run off the end!! ***" << endl;
                        return ;
                }
                (_commandList[_numCommands]→_receiver→
                        *(_commandList[_numCommands] ->_action))();
                _numCommands = _numCommands + 1;
        }

protected:
          Number *_receiver;
          Action _action;
          static Command *_commandList[20];
          static Memento *_mementoList[20];
          static int _numCommands;
          static int _highWater;

}; // END class... Command
```

Continued ...

```
// INITIALIZATION
Command *Command::_commandList[];
Memento *Command::_mementoList[];
int Command::_numCommands = 0;
int Command::_highWater = 0;

// ===================================
// THE MAIN PROGRAM
int main() {
        int i;
        cout << "Integer: ";
        cin >> i;
        Number *object = new Number(i);

        Command *commands[3];
        commands[1] = new Command(object, &Number::dubble);
        commands[2] = new Command(object, &Number::half);

        cout << "Exit[0], Double[1], Half[2], Undo[3], Redo[4]: ";
        cin >> i;

      while (i) {
            if (i == 3)
                  Command::undo();
            else if (i == 4)
                  Command::redo();
            else
                  commands[i]→execute();
                  cout << "   " << object->getValue() << endl;
                  cout << "Exit[0], Double[1], Half[2], Undo[3], Redo[4]: ";
                  cin >> i;
      }// END while

} // END main()
```

Fig: Source code for Question B3(3)

End of Page